EXHIBIT Z

TO DECLARATION OF S. MERRILL WEISS IN SUPPORT OF PLAINTIFF ACACIA MEDIA TECHNOLOGIES CORPORATION'S MEMORANDUM OF POINTS AND AUTHORITIES IN OPPOSITION TO ROUND 3 DEFENDANTS' MOTION FOR SUMMARY JUDGMENT OF INVALIDITY UNDER 35 U.S.C. § 112 OF THE '992, '863, AND '702 PATENTS; AND SATELLITE DEFENDANTS' MOTION FOR SUMMARY JUDGMENT OF INVALIDITY OF THE '992, '863, AND '720 PATENTS

Hovsepian v. Apple, Inc. Doc. 325 Att. 27

Dockets.Justia.com

Neural Networks, Vol. 3, pp. 277–290, 1990
Printed in the USA. All rights reserved.
0893-6080/90 $3.00 + .00
Copyright © 1990 Pergamon Press plc

ORIGINAL CONTRIBUTION

Competitive Learning Algorithms for Vector Quantization

STANLEY C. AHALT, ASHOK K. KRISHNAMURTHY, PRAKOON CHEN, AND DOUGLAS E. MELTON

Ohio State University, Columbus

(*Received 2 December 1988; revised and accepted 27 September 1989*)

Abstract—*We compare a number of training algorithms for competitive learning networks applied to the problem of vector quantization for data compression. A new competitive-learning algorithm based on the "conscience" learning method is introduced. The performance of competitive learning neural networks and traditional non-neural algorithms for vector quantization is compared. The basic properties of the algorithms are discussed and we present a number of examples that illustrate their use. The new algorithm is shown to be efficient and yields near-optimal results. This algorithm is used to design a vector quantizer for a speech database. We conclude with a discussion of continuing work.*

Keywords—Neural networks, Vector quantization, Encoding, Speech.

I. INTRODUCTION

A vector quantizer (VQ) statistically encodes data vectors in order to quantize and compress the data. Even in those systems in which substantial channel bandwidth is available, such as those employing fiber optic communication links, there is still a need to compress data. Vector quantization has been shown to be useful in compressing data that arises in a wide range of applications, including image processing (Nasrabadi & King, 1988; Gray, 1984), speech processing (Makhoul, Roucos, & Gish, 1985), facsimile transmission (Netravali & Mounts, 1980), and weather satellites (Elachi, 1987). Typically, the applications which employ VQ require large amounts of storage or transmission bandwidth, and can tolerate some loss of fidelity for the sake of compression.

From rate distortion theory, it can be shown that vector quantization can achieve better compression performance than any conventional coding technique which is based on the encoding of scalar quantities (Gray, 1984). Because of this, there has been extensive research into VQ. However, practical use of VQ techniques has been limited because of the prohibitive amount of computation associated with existing encoding algorithms (Davidson, Cappello, & Gersho, 1988).

We discuss the application of competitive learning (CL) neural networks to VQ. A new CL training algorithm, which represents one method of embodying the conscience principle enunciated by Grossberg (Grossberg, 1976a, 1976b, 1987), is developed for designing VQ codebooks. This algorithm yields near-optimal results and can be used to develop adaptive vector quantizers. It is also computationally more efficient than other neural network vector quantizers which have appeared in the literature.

We first describe the *basic* memoryless vector quantizer. We then discuss various training algorithms for neural network vector quantizers, including the new training algorithm that we have developed. Applications of our technique follow, and we conclude with a discussion of our work in progress.

II. VECTOR QUANTIZATION

Gray (1984) gives a complete introduction to VQ, and Gersho (1982) presents a thorough discussion of the theory of vector quantization. Lloyd (1982) discusses an algorithm for optimal quantization of data, and this algorithm is extended by Linde, Buzo, and Gray (1980) to arbitrary distortion measures and to vectors. The Linde, Buzo, and Gray (LBG) algorithm is widely used to design the codebooks for VQ. Finally, a discussion of the use of vector quantizers in speech applications can be found in (Makhoul et

This research was partially supported by a Cray University Research and Development Grant and by the GE Young Faculty Program.

Address inquiries to Stanley C. Ahalt, Department of Electrical Engineering, Ohio State University, Columbus, OH 43210.

al., 1985) and its use in image coding is described in (Nasrabadi & King, 1988).

While a detailed theoretical discussion of vector quantization is beyond the scope of this paper, we briefly discuss the basic concepts below.

A. Basic Vector Quantization Concepts

Vector quantization is a quantization technique which capitalizes on any underlying structure in the data being quantized. The space of the vectors to be quantized is divided into a number of regions and a reproduction vector is calculated for each region. Given any data vector to be quantized, the region in which it lies is determined and the vector is represented by the reproduction vector for that region. Instead of transmitting or storing a given data vector, a symbol which indicates the appropriate reproduction vector is used. This can result in considerable savings in transmission bandwidth, albeit at the expense of some distortion.

More formally, vector quantization maps arbitrary data vectors to a binary representation or symbol. Thus, the VQ mapping is from a k-dimensional vector space to a finite set of symbols $\mathbf{M}$. Associated with each symbol $m \in \{\mathbf{M}\}$ is a reproduction vector $\hat{\mathbf{x}}_m$. The encoding of the data vector $\mathbf{x}$ to the symbol m is a mapping,

$$\mathrm{VQ} : \mathbf{x} = (x_1, x_2, \ldots, x_k) \longrightarrow m,$$

where $m \in \{\mathbf{M}\}$ and the set $\mathbf{M}$ has size M. Assuming a noiseless transmission or storage channel, m is decoded as $\hat{\mathbf{x}}_m$, the reproduction vector associated with the symbol m. The collection of all possible reproduction vectors is called the *reproduction alphabet* or more commonly the *codebook*. Since there are M elements in the set $\mathbf{M}$, there are M possible entries in the codebook. Once the codebook is constructed and, if necessary, transmitted to the receiver, the encoded symbol m acts as an index into the codebook. Thus, the *rate* R, of the quantizer is $R = \log_2 M$ in bits per input vector. Since each input vector has k components, the number of bits required to encode each input vector component is R/k.

Since each data vector must be ultimately represented as one of the codebook entries, the composition of the codebook determines the overall performance of the system. A number of different performance criteria can be used to determine an optimal codebook. For example, in speech and image transmission applications, the usual objective is to minimize the overall distortion in the signal due to VQ. Thus the design criterion used to design an optimal codebook is the minimization of the average distortion in encoding vectors using the codebook. Another possible criterion is to maximize the entropy of the codebook, i.e., ensure that each of the codewords is used equally frequently in encoding the data. The idea here is to ensure that all the codewords are doing their "fair share" in representing the input data. As we shall see, this is a very useful criterion in developing neural network training algorithms for VQ design. In general, these two criteria are not equivalent; however, it can be shown that for the case when M is fixed and k is very large, the codebook which maximizes entropy also minimizes the expected distortion (Makhoul et al., 1985; Shannon, 1960).

Given a performance criterion, the VQ design process involves the determination of a codebook that is optimal with respect to this criterion. In general, this requires knowing the probability distribution of the input data. Typically, however, this distribution is not known, and the codebook is constructed through a process called *training*. During training, a set of data vectors that is representative of the data that will be encountered in practice is used to determine an optimal codebook.

During the training process, a distortion measure $d(\mathbf{x}, \hat{\mathbf{x}})$ is typically used to determine which data points are to be considered as being in the same region. The distortion measure can be viewed as the cost of representing $\mathbf{x}$ as $\hat{\mathbf{x}}$. By determining which training data vectors lie in the same region, the k-dimensional data space is partitioned into cells. All of the input vectors that fall into a particular cell are mapped to a single, common reproduction vector. If the cells are partitioned according to a minimum distortion rule, then the partition is referred to as a Voronoi or Dirichlet partition.

As can be seen, the distortion measure plays a major role in determining the effectiveness of the vector quantizer in particular applications. Examples of some common distortion measures are:

- *Euclidean distance*:

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \sqrt{\sum_{i=1}^{k} (x_i - \hat{x}_i)^2}$$

- *Itakura–Saito distortion*: The Itakura–Saito (IS) distortion measure is frequently used in speech coding applications, and is a spectral distortion measure. Let $X(\omega)$ and $Y(\omega)$ represent the spectrum of short-time segments of two speech signals. The IS distortion between the two spectra is defined as

$$d(X(\omega), Y(\omega)) = \int_{-\pi}^{\pi} \left[\frac{X(\omega)}{Y(\omega)} + \ln\left(\frac{Y(\omega)}{X(\omega)}\right) - 1 \right] \frac{d\omega}{2\pi}.$$

In many applications where VQ is used for speech coding, the short-time spectrum of the speech signal is obtained as an all-pole model using linear prediction analysis (O'Shaughnessy, 1987), i.e.,

$$X(\omega) = \frac{\sigma_X^2}{|1 - \sum_{i=1}^{k} a_{iX} e^{-j\omega i}|^2},$$

$$Y(\omega) = \frac{\sigma_Y^2}{|1 - \sum_{i=1}^{k} a_{iY} e^{-j\omega i}|^2}.$$

It can be shown that for such all-pole models, the IS distortion can be written as (O'Shaughnessy, 1987)

$$d(X(\omega), Y(\omega)) = \frac{\mathbf{a}_Y^t R_X \mathbf{a}_Y}{\sigma_Y^2} + \ln\left(\frac{\sigma_Y^2}{\sigma_X^2}\right) - 1, \qquad (1)$$

where R_X is the $(k+1)X(k+1)$ matrix formed from the autocorrelations of the signal $X(\omega)$, and $\mathbf{a}_Y = (1 a_{1Y} \cdots a_{kY})^t$ is a length $(k+1)$ column vector. From (1), it is clear that the IS distortion measure is very similar to a weighted inner product, with the weighting matrix R_X being signal dependent. Also note that the IS distortion is not symmetric, i.e.,

$$d(X(\omega), Y(\omega)) \neq d(Y(\omega), X(\omega)). \qquad (2)$$

In order to minimize the amount of computation, the IS distortion is usually computed as (Buzo, Gray, Gray, & Markel, 1980)

$$d(X(\omega), Y(\omega)) = b_Y(0) r_X(0) + 2 \sum_{i=1}^{k} b_Y(i) r_X(i), \qquad (3)$$

where $r_X(0) \cdots r_X(k)$ are the autocorrelations of the signal $X(\omega)$ and

$$b_Y(i) = \sum_{j=0}^{k-i} a_{jY} a_{j+i,Y}, \quad a_{0Y} = 1.$$

The training process which is used to build the codebook can be summarized as follows. Each of the data vectors is compared to each of the codewords, and the corresponding distortion is calculated. The codeword that most closely matches the data vector, i.e., the reproduction vector which represents the input vector with the minimum distortion, is selected and the codeword is modified to reflect the inclusion of this new data vector in its partition.

After the codebook has been constructed, input data vectors are encoded in a similar manner. Each data vector is compared with each of the codewords, and the index of the codeword which yields the minimum distortion is transmitted or stored. When the data is decoded, the codebook uses the index to access the reproduction vector which is then used to represent the original data.

As can be seen from the above discussion, the training and encoding processes are computationally expensive. Moreover, most of the algorithms currently used for VQ design such as the LBG algorithm are batch mode algorithms (Linde et al., 1980), and need to have access to the entire training data set during the training process. Given the large training data sets required to form an adequate representation of the input vector space in some applications (e.g., speaker independent speech coding), the batch mode training algorithms appear infeasible using existing computer technology. Also, in many communication applications, changes in the communication channels means that a codebook designed under one condition is inappropriate for use in another condition. Under these circumstances (both for handling large training data sets and changing channel conditions), it is much more appropriate to work with adaptive VQ design methods, even if these are suboptimal in a theoretical sense. By an adaptive VQ, we mean that the codebook vectors are changed with the arrival of each new training vector, and no "batching" of the training data vectors is essential. A major advantage of formulating VQ as neural networks is that the large number of adaptive training algorithms that are used for neural networks can now be applied to VQ.

In the next section we discuss some of the neural network vector quantizers that have appeared previously in the literature, and analyze our algorithm with respect to the previous work.

III. NEURAL NETWORKS FOR VECTOR QUANTIZATION

A number of researchers have demonstrated the use of neural network techniques in VQ encoding and codebook design (training). For example, a number of researchers (Naylor & Li, 1988) have used Kohonen self-organizing feature maps (Kohonen, 1984) to construct VQ codebooks for speech applications, while (Nasrabadi and Feng, 1988) used them to build VQ codebooks for image coding. Similarly, work with variable region vector quantization of both speech and image is discussed in (Matsuyama, 1988). A brief review of the general framework of neural network vector quantization is given below.

A. Encoding

It is straightforward to formulate a neural network structure for the encoding of vectors. As before, let the vectors which are to be quantized be from a k-dimensional vector space, and let a distortion measure $d(\mathbf{x}, \mathbf{y})$ be defined in this space. Let the size of the codebook be M, and let the codewords be $\mathbf{c}_i$, $i = 1, \ldots, M$. Consider a neural network with M neural units, and make the ith codeword $\mathbf{c}_i$ the weight vector associated with neural unit i. Given any vector $\mathbf{x}$ that is to be encoded, $\mathbf{x}$ is fed in parallel to all the M neural units. Each of these units computes the distortion between the input vector and its weight vector,

$$d_i = d(\mathbf{x}, \mathbf{c}_i), \quad i = 1, \ldots, M. \tag{4}$$

The input vector is then encoded as the *index* i^* of the neural unit with the minimum distortion,

$$d_{i^*} = \min_i d_i. \tag{5}$$

For a specific codebook, the "nearest-neighbour" encoding procedure described above is optimal (Gray, 1984). Note that all computations, except for picking the "winning" neural unit and determining its index, are carried out in parallel. A number of neural network methods for picking the winner have been discussed in the literature. These include Grossberg's on-center, off-surround method (Grossberg, 1976a, 1976b), Lippman et al.'s MAXNET (Lippmann, 1987) and the minimum distance automaton of Winters and Rose (1989), which will be discussed in greater detail. Also, as shown by Hecht-Nielsen (1988), it possible to directly compute the index number of the winning neural unit by adding an additional output layer to the neural network.

B. Training

We have shown in the previous section that neural networks can be used for encoding in vector quantizers. A more fundamental benefit of formulating vector quantization as a neural network task is that the large body of neural network training algorithms that have been developed can now be adapted to the problem of training vector quantizers. We discuss below three alternative training algorithms: the competitive learning (CL) network, the Kohonen self-organizing feature map (KSFM), and the frequency-sensitive competitive learning (FSCL) network.

1. *The competitive learning network.* An adaptive version of the LBG algorithm for the training of vector quantizers can be accomplished with a competitive learning (CL) network. Assume that the neural network VQ is to be trained on a large set of training data. Further assume that the M neural units are initialized with the weight vectors $\mathbf{w}_i(o)$, $i = 1, \ldots, M$. These weights can be generated randomly or they can be the first M vectors of the training set.

Starting with these initial weight vectors, the training algorithm iterates a number of times through the training data, adjusting the weight vectors of the neural units after the presentation of each training vector. The algorithm used to adjust the weight vectors is based on competitive learning. Competitive learning has been studied in depth by Grossberg (1976a, 1976b, 1987) and Kohonen (1984, 1988c, 1988a, 1988b) and by other researchers (Rumelhart & Zipser, 1985; Rumelhart & McClelland, 1986; Hecht-Nielsen, 1988; Steels, 1988).

The algorithm for updating the weight vectors is as follows. The input vector $\mathbf{x}$ is presented to all of the neural units and each unit computes the distortion between its weight and the input vector. The unit with the smallest distortion is designated as the winner and its weight vector is adjusted towards the input vector.

Let $\mathbf{w}_i(n)$ be the weight vector of neural unit i before the input is presented. The output z_i of the unit is computed as follows.

$$z_i = \begin{cases} 1 \text{ if } d(\mathbf{x}, \mathbf{w}_i(n)) \le d(\mathbf{x}, \mathbf{w}_j(n)), \ j = 1, \ldots, M \\ 0, \text{ otherwise.} \end{cases} \tag{6}$$

The new weight vectors $\mathbf{w}_i(n + 1)$ are computed as

$$\mathbf{w}_i(n + 1) = \mathbf{w}_i(n) + \varepsilon(\mathbf{x} - \mathbf{w}_i(n))z_i. \tag{7}$$

In the above equation, the parameter ε is the learning rate, and is typically reduced monotonically to zero as learning progresses.

A problem with this training procedure is that it sometimes leads to neural units which are underutilized. An excellent discussion of this problem can be found in (Grossberg, 1976a, 1976b) and subsequently in (Rumelhart & Zipser, 1985; Grossberg, 1987). Other researchers have used various methods to address this problem (Hecht-Nielsen, 1988; DeSieno, 1988; Van den Bout & Miller, 1989).

The following example illustrates the problem of node underutilization. The data for this example is from four clusters. A neural network with four units was trained using the above procedure. The initial weight vector of all the units was set to the same value of (1, 0). As shown in Figure 1, only one of the weight vectors associated with the neural units is modified during training. The index of this single neural unit is consequently used for the quantization of data from all four clusters, resulting in an artificially high distortion rate. Note that typically the initial weight vectors for CL training are picked randomly; we have chosen to pick them all to be the same value as an extreme example to illustrate the underutilization problem of CL networks.

2. *The Kohonen self-organizing feature map.* Another neural network structure that has been used for vector quantization is the Kohonen self-organizing feature map (KSFM) (Kohonen, 1984; Naylor & Li, 1988). It should be noted, however, that Kohonen has primarily focused the use of KSFM on pattern classification applications and not for general signal representation. Kohonen initially conceived of

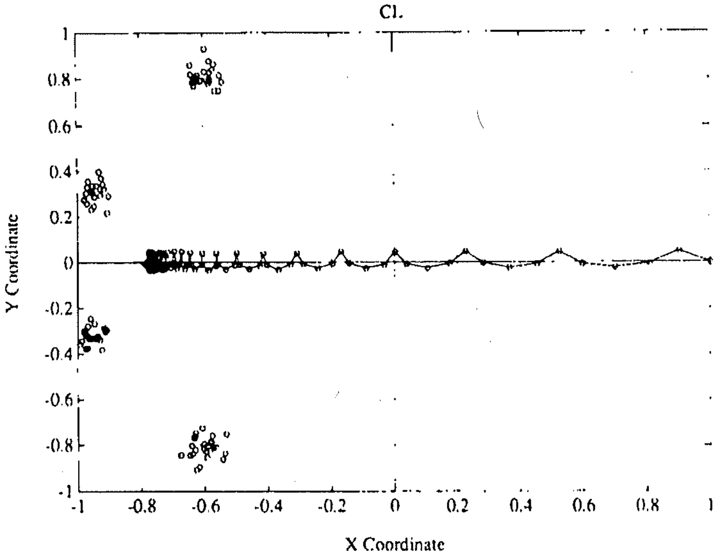


FIGURE 1. Movement of references vectors, four data clusters, competitive learning.

the KSFM network structure to illustrate the formation of topological feature maps in the brain. The KSFM and the competitive learning network are similar; however, in the KSFM structure each neural unit has an associated topological neighbourhood of other neural units. During the training process, the winning neural unit as well as the neural units in the neighbourhood of the winner is updated. The size of the neighbourhood is decreased as training progresses until each neighbourhood has only one unit, i.e., the KSFM net becomes a CL net after sufficient training.

Let $\mathbf{w}_i(n)$ be the weight associated with the ith neural unit, and let $\mathbf{x}$ be the input vector. Compute the distortion $d(\mathbf{x}, \mathbf{x}_i(n))$, $i = 1, \ldots, M$, and let the neural unit with the minimum distortion be i^*. Also, let $N(i^*)$ be the topological neighbourhood associated with unit i^*. The weight update equations are:

$$\mathbf{w}_i(n+1) = \begin{cases} \mathbf{w}_i(n) + \varepsilon(n)[\mathbf{x} - \mathbf{w}_i(n)], & i \in N(i^*) \\ \mathbf{w}_i(n), & \text{otherwise.} \end{cases} \tag{8}$$

Note that the gain sequence $\varepsilon(n)$ slowly decreases with time to zero.

As can be seen from these equations the KSFM structure involves more computation than the competitive learning network. At each step of the training process, the neighbourhood $N(i^*)$, of the winning node must be computed and all of the units in the neighbourhood updated.

By the use of neighborhoods, the KSFM network overcomes the problem of underutilized nodes discussed above. In Figure 2 we show the behavior of the KSFM for the four cluster problem discussed earlier. As shown in Figure 2, all four neural units learn, and the weight vectors move to the center of the four data clusters. Note that this example clearly illustrates the two phase learning mentioned by Kohonen (1984). All four weight vectors initially move together to the center of all four clusters of data. Then, once the neighbourhood size is reduced to one, convergence to the data cluster centroids occurs.

One drawback of the KSFM network compared to the competitive learning network is the additional computation involved during training. This additional computation arises from both the calculation of the neighbourhood of the winning unit, and from the updating of all members of the neighbourhood. We describe below an alternative neural network that does not require the calculations associated with neighborhoods and solves the CL problem of underutilized neural units.

3. *Frequency-sensitive competitive learning.* One of the motivations for the frequency-sensitive competitive learning (FSCL) network is to overcome the limitations of the simple competitive learning network while retaining its computational advantages. Since one of the main problems with the CL network is that some of the neural units may be underutilized, the learning algorithm for the FSCL network keeps a count of how *frequently* each neural unit is the winner. This information is used to ensure that, during the course of the training process, all neural units are modified an approximately equal number of times.

A similar approach was first described by Grossberg (1976a, 1976b, 1987), who suggested the use of a variable-threshold model to overcome the prob-

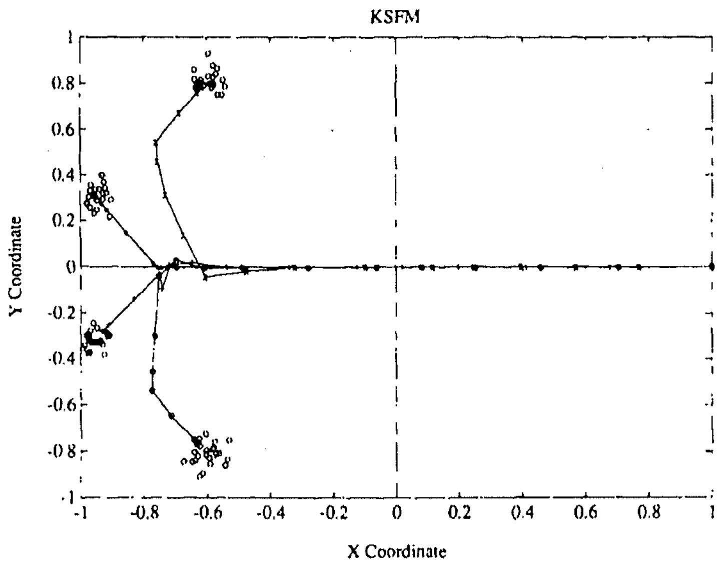


FIGURE 2. Movement of references vectors, four data clusters, Kohonen self-organizing feature maps.

lem. In the Grossberg model a neural unit becomes increasingly sensitive when if fails to win competitions, and less sensitive when winning frequently. Additionally, the learning rate is also varied according to winning, i.e., inactivity increase both the likelihood of winning and the learning rate and vice versa. Another similar approach, called the conscience method, has been suggested by DeSieno (DeSieno, 1988; Hecht-Nielsen, 1988) and subsequently used in slightly modified form in (Van den Bout & Miller, 1989).

The FSCL method represents a method of applying the models introduced by Grossberg to the problem of vector quantization. In the FSCL network,

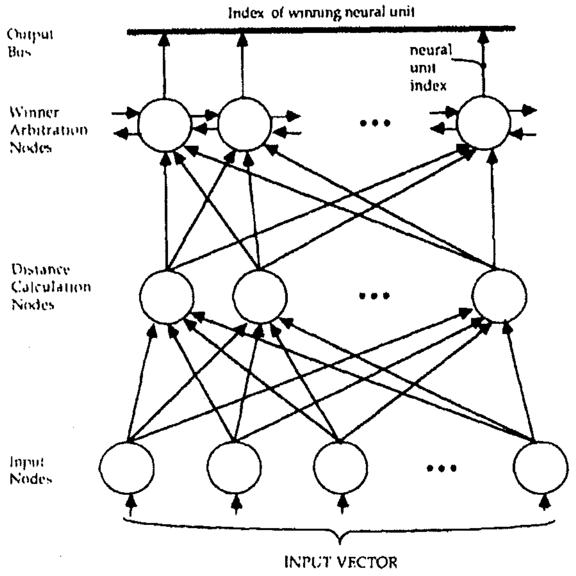


FIGURE 3. An implementation of the FSCL network.

each unit incorporates a count of the number of times it has been the winner. The distortion measure used to determine the winner is directly modified to include this count. Specifically, let $d(\mathbf{x}, \mathbf{w}_i(n))$ be the distortion measure that we are interested in minimizing during the quantization process, and let $u_i(n)$ be the total number of times that neural unit i has been the winner during training. A modified distortion measure for the training process is defined as:

$$d^*(\mathbf{x}, \mathbf{w}_i) = d(\mathbf{x}, \mathbf{w}_i(n)) \cdot u_i(n). \tag{9}$$

The winning neural unit at each step of the training process is the unit with the minimum d^*.

Note that if a given neural unit wins the competition frequently, its count and consequently d^* increase. This reduces the likelihood that this unit will be the winner, giving other units with a lower count value a chance to win the competition. However, no change is made to the learning rate.

The computations implied by (9) can be implemented using only local operations in a number of ways. A possible optical implementation is described in (Ahalt, Collins, Krishnamurthy, & Stewart, 1989). We illustrate an electronic VLSI implementation based on (Winters & Rose, 1989) in Figure 3. The network shown consists of three layers: an input layer that distributes the input vector to the second layer units; a second layer of units, where each unit computes the modified distortion d^* between its weight vector and the input; and an output layer based on the minimum distance automaton (Winters & Rose, 1989) that determines the winning neural unit from the distortions computed by the second layer units. Thus, there are k units in the input layer, and M units in the middle and output layers, where k is the dimension of the input vector, and M is the size of

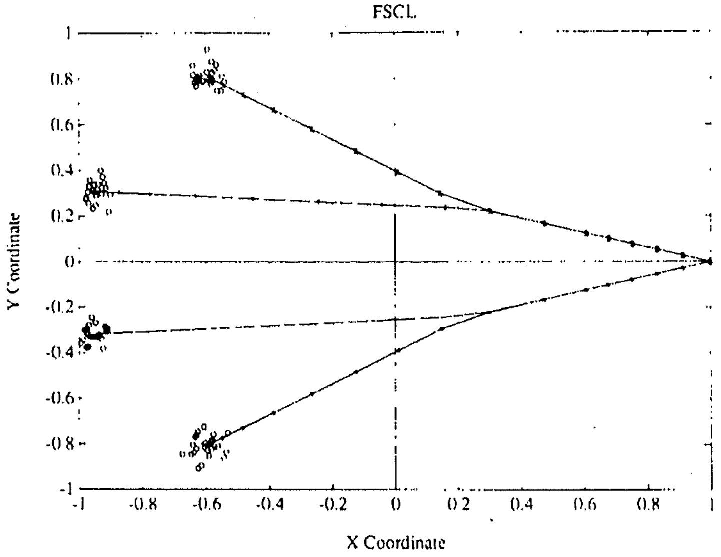


FIGURE 4. Movement of references vectors, four data clusters, frequency sensitive competitive learning.

the codebook. As shown by Winters and Rose (1989), once the distortions have been computed by the middle layer, the output layer can determine the winning unit in at most $\log_2 M$ clock cycles. Once the winning unit is determined, this information is used by winning neural unit to update its count $u_i(n)$, as well as its weight vector $\mathbf{w}_i(n)$.

In Figure 4 we illustrate the performance of the FSCL network on the same four cluster problem described previously. As can be seen from the figure, all four neural units are utilized and they move more directly to the cluster centers than the KSFM network.

Thus, in the FSCL network, each neural unit dynamically adapts its chances of winning based on *both* the distortion and the number of times it has been modified. This property ensures that the neural units will be modified an (approximately) equal number of times.

It is interesting to compare the performance of the FSCL method to other CL algorithms which incorporate "conscience" in the training process. An example is DeSieno's conscience method which is designed to form a nonparametric model of the underlying probability density function of the input data. In DeSieno's model, each neural unit monitors its success and a bias is developed based on the number of times a neural unit wins the competition. Units which win substantially more often than other neural units temporarily withdraw from the competition, allowing other neural units to be modified. The performance of this method was demonstrated on two one-dimensional problems (DeSieno, 1988), and the results show that the conscience method results in a relatively equiprobable distribution of the exemplar vectors.

However, when applied to the VQ problem, DeSieno's method does not necessarily yield the lowest distortion. This can be demonstrated using the example discussed in (DeSieno, 1988). Figure 5 shows two probability functions. Figure 5(a) shows the density function generated from the product of two uniformly distributed random variables between 0.0 and 1.0, and Figure 5(b) the shows the same distribution with all values between 0.2 and 0.4 removed. In Figure 6 we show the resulting weight values developed using the FSCL training algorithm using 15 processing elements. The FSCL algorithm yields weights similar to those reported by DeSieno for both these distributions. However, Table 1 shows that the FSCL training method yields lower distortion for both distributions.

IV. VECTOR QUANTIZATION OF SPEECH

In this section we compare the performance of the vector quantization methods that we have discussed earlier when used to encode speech data from a single male speaker. The vectors used for training and encoding were the ten autocorrelation coefficients (O'Shaughnessy, 1987) obtained from short-time windows of the speech signal. In voiced speech segments, the autocorrelation coefficients were computed over one pitch period long windows of the speech signal, i.e., the autocorrelation coefficients were computed pitch-synchronously. The pitch periods were determined using the residual signal from linear prediction (LP) analysis (Entropic Speech, Inc., 1989). In unvoiced segments, the autocorrelation coefficients were computed over 20 ms windows, with successive windows being overlapped by 10 ms. The sampling rate was 10 KHz.

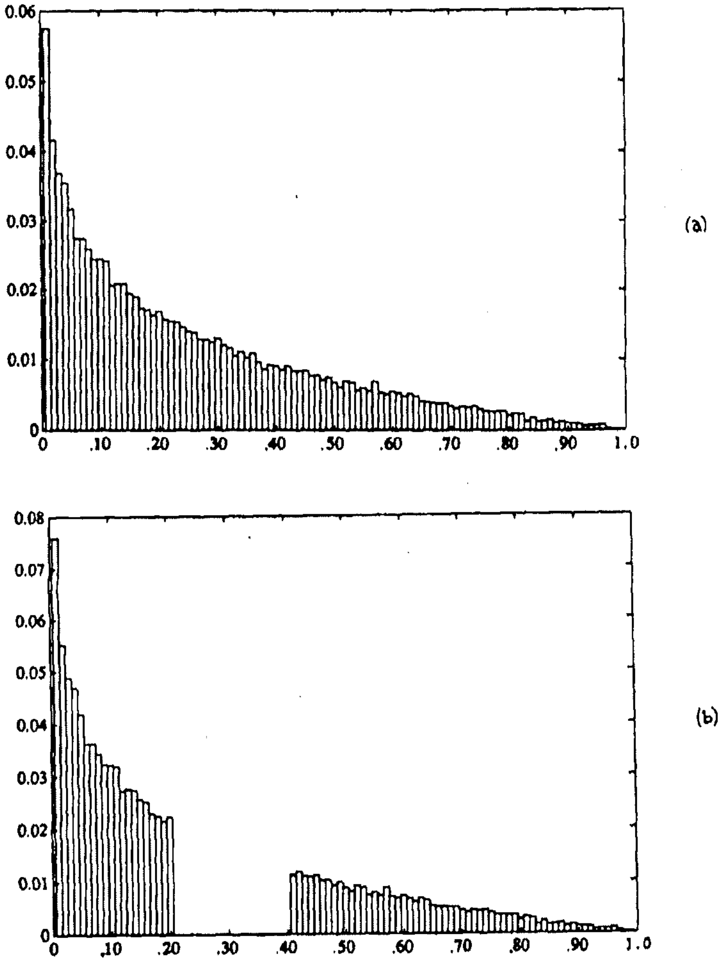


FIGURE 5. Probability density functions: (a) product of two uniform random variables between 0 and 1, (b) same distribution with a gap introduced between 0.2 and 0.4.

The training data was obtained from the analysis of 24 sentences of approximately 3 s duration each. This resulted in a training set of about 14,000 ten-dimensional vectors, with each vector being the ten autocorrelation coefficients for an analysis frame. The IS distortion measure described in Section II was used during training and encoding. The form of the IS distortion as shown in eqn (2) was used for the computation. During both training and testing, each codeword was stored in the form $(r_X(0)r_X(1)r_X(2) \cdots r_X(k))^T$. In the training data file, each training datum was stored in two equivalent forms: as the vectors $(b_Y(0)2b_Y(1) \cdots 2b_Y(k))^T$ and $(r_Y(0)r_Y(1) \cdots r_Y(k))^T$. This allows the distortion for each neural unit to be computed using eqn (2) using the first form of the training vector. Once the winning unit is determined, the weight vector can be updated using eqn (7) and the second form of the training vector. Storing both forms of the training datum thus allows us to use the simple inner product form of the IS distortion measure, and also allows us to update the weight vectors easily, without the additional computation needed to change between the two forms.

Figures 7 and 8 show the average distortion in encoding the training data set as a function of the size of the codebook. The results are shown for codebook sizes of 4, 16, 32, 64, and 128 vectors. For the neural network VQs, the initial weight vectors for a codebook of size N were chosen as the first N vectors in the training set. For the KSFM method, the neighborhoods were chosen as arbitrary rectangular regions; for example, for the 128 size codebook, it was chosen as a 16×8 rectangular array. Figure 7 shows that the performance of the CL method is much poorer than that of the other training methods, particularly when the codebook size is small. In Figure 8 we have reduced the scale for the average distortion to allow a better comparison of methods other than CL. This figure reveals that the performance of LBG method and the FSCL net are very close when the

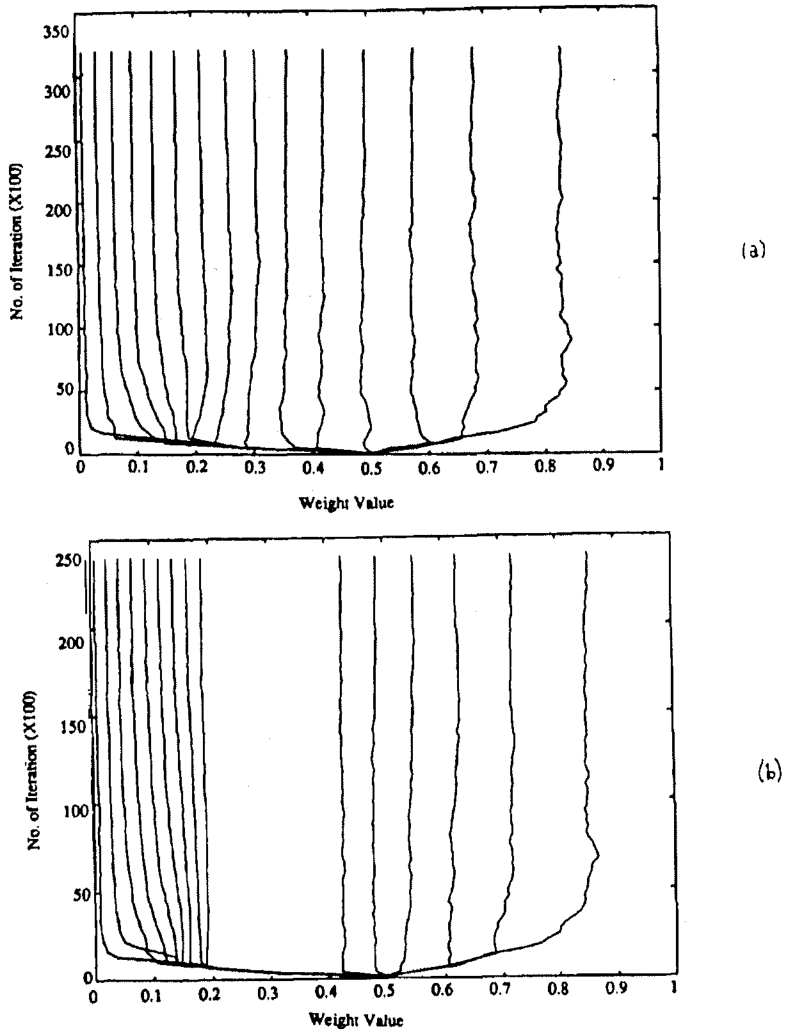


FIGURE 6. Weight values of the FSCL method for probability density functions of Figure 5.

codebook size is 32 or larger. The KSFM method leads to a higher average distortion for all of the codebook sizes studied.

In Figures 9–12 we show the codeword utilization for each of the methods for a codebook size of 128. Note that the FSCL method leads to the most uniform utilization of the codewords in the sense that each codeword is used approximately an equal number of times. For each method, the figures also show the average Itakura–Saito distortion and the entropy

TABLE 1
Comparative Distortion

Distribution	Average square error	
	Conscience	FSCL
Without gap—Figure 5(a)	7.005×10^{-4}	3.549×10^{-4}
With gap—Figure 5(b)	4.583×10^{-4}	2.516×10^{-4}

of the codebook. The entropy is computed as

$$E = -\sum_{i=1}^{128} p_i \log_2(p_i), \tag{10}$$

where p_i is the relative frequency with which codeword i was used in encoding the data set. Note that in the ideal case, where all codewords are utilized equally often, the value of the entropy is 7.

The CL method utilizes one codeword 40% of the time and many of the neural units are never used to encode the data. The KSFM codebook shows greater uniformity in codebook utilization and results in an entropy of 6.574. This is consistent with the results reported by Naylor and Li (1988) which show that the KSFM technique yields codebooks of more uniform utilization than those produced by covering followed by K-means clustering. The FSCL network not only yields the highest entropy, 6.917, but also the lowest Itakura–Saito distortion among all of the

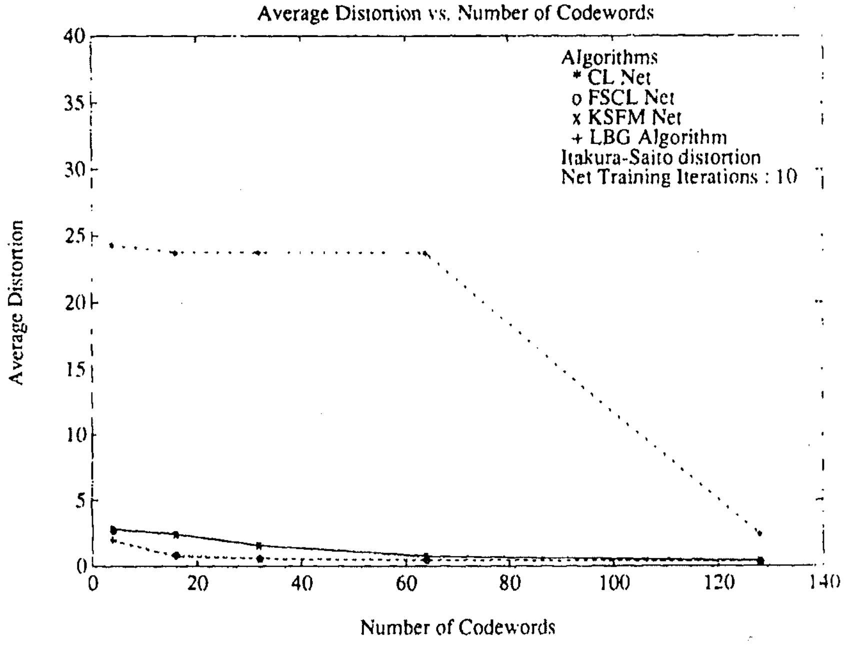

Average Distortion vs. Number of Codewords
Algorithms
* CL Net
o FSCL Net
x KSFM Net
+ LBG Algorithm
Itakura-Saito distortion
Net Training Iterations : 10
Average Distortion
Number of Codewords
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
120
140

FIGURE 7. Average distortion vs. number of codewords.

methods. Thus, for this realistic application, the FSCL method yields a codebook design that is most comparable to the codebook which is produced with the optimal LBG method.

In Figure 13 we show the effects of the learning step-size ε. This figure compares the average Itakura–Saito distortion in encoding the training set as a function of the iteration number during training. Three cases are compared: the FSCL method with a fixed ε (ε = 0.01), the FSCL method with a decreasing ε ($\varepsilon = 0.01e^{-0.239(n-1)}$, where n is the training epoch number), and the KSFM method with a decreasing ε ($\varepsilon = 0.01e^{-0.239(n-1)}$, where n is the training epoch number). Note that for this particular choice of ε the FSCL method appears to learn much faster than the KSFM method. Although the FSCL method

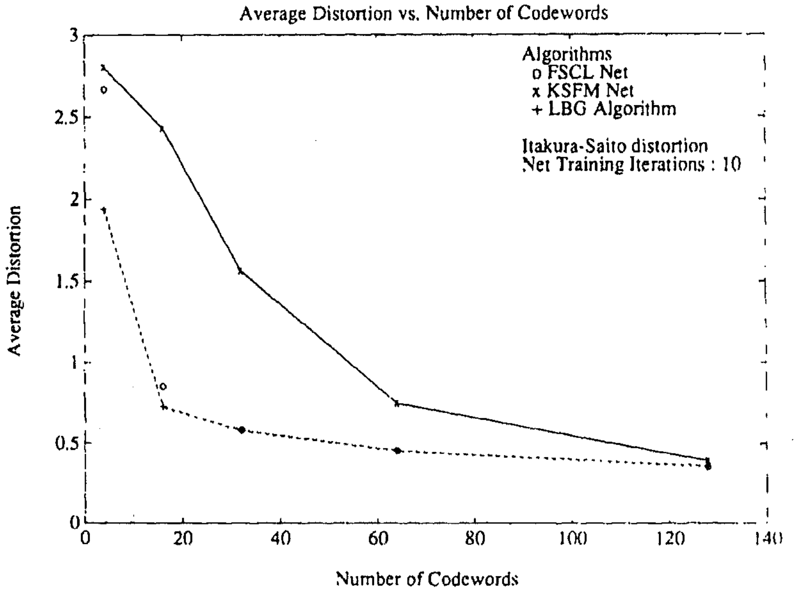

Average Distortion vs. Number of Codewords
Algorithms
o FSCL Net
x KSFM Net
+ LBG Algorithm
Itakura-Saito distortion
Net Training Iterations : 10
Average Distortion
Number of Codewords
0
0.5
1
1.5
2
2.5
3
0
20
40
60
80
100
120
140

FIGURE 8. Average distortion vs. number of codewords.

Exhibit Z Page 506

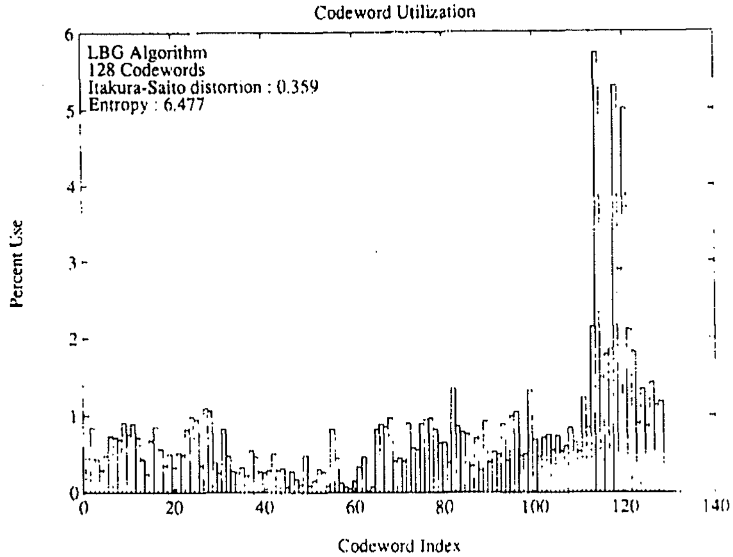

Codeword Utilization
LBG Algorithm
128 Codewords
Itakura-Saito distortion : 0.359
Entropy : 6.477
Percent Use
Codeword Index
0
1
2
3
4
5
6
20
40
60
80
100
120
140

FIGURE 9. Codeword utilization, LBG algorithm.

with the fixed ε leads to a lower average distortion at the end of 30 iterations, the distortion increases around iteration 15 and then slowly decreases. We conjecture that in this case the behavior of the average distortion represents a form of "convergence in the mean". We are presently investigating the effects of using a fixed and variable ε on the rate of convergence.

In Figure 14 we compare the spectrogram of the sentence, "The birch canoe slid on the smooth planks" with the spectrogram of the synthesized version of the same sentence, Figure 15. The synthesis was done from the vector-quantized autocorrelation parameters, with a codebook size of 128. We have also synthesized other sentences, both using data which was part of the training material, and data collected from the speaker at a later time and not used in the training of the VQs. Informal listening tests of the original speech, the synthesized speech without VQ, and the vector-quantized and synthesized speech show that while there is a quality decrease between the original and the synthesized speech, there is no significant additional degradation due to the vector quantization step.

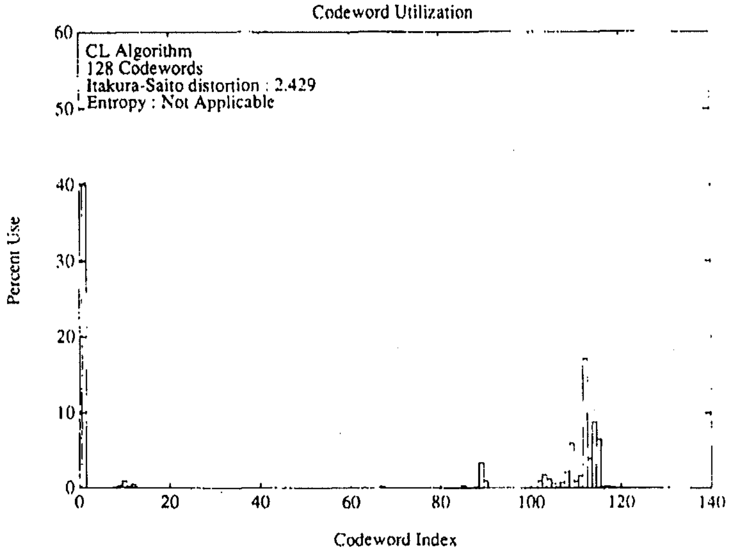
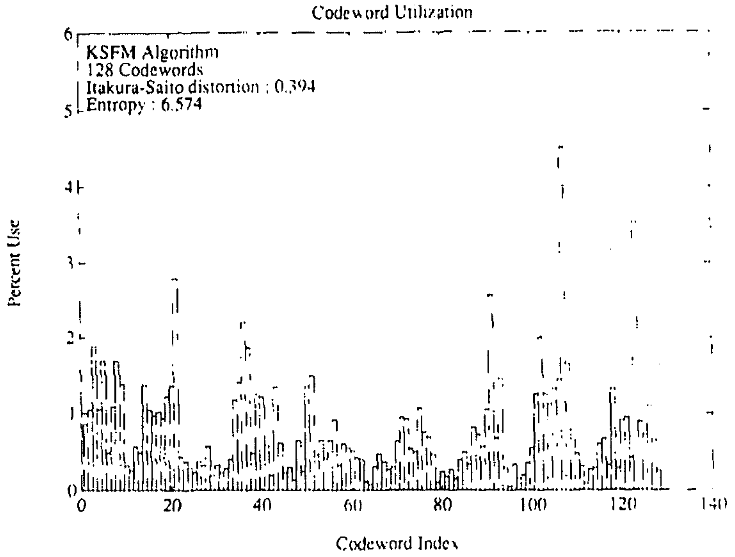

Codeword Utilization
CL Algorithm
128 Codewords
Itakura-Saito distortion : 2.429
Entropy : Not Applicable
Percent Use
Codeword Index
0
10
20
30
40
50
60
20
40
60
80
100
120
140

FIGURE 10. Codeword utilization, competitive learning net.



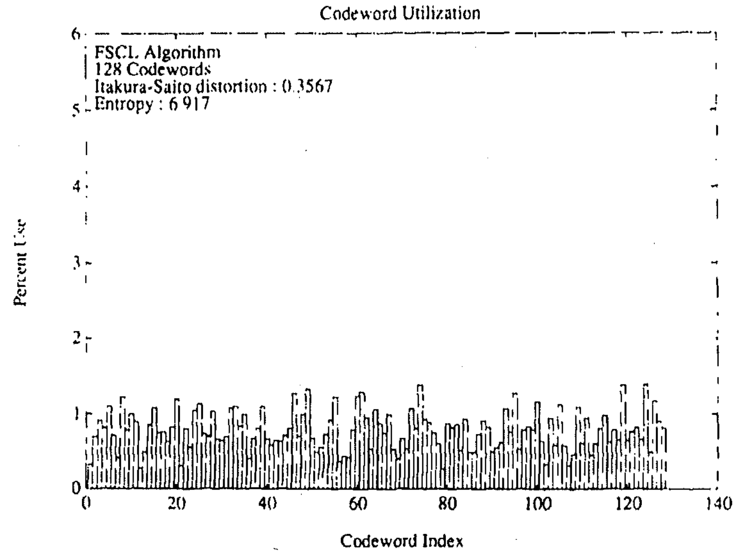
Codeword Utilization
KSFM Algorithm
128 Codewords
Itakura-Saito distortion : 0.394
Entropy : 6.574
Percent Use
Codeword Index
0
1
2
3
4
5
6
20
40
60
80
100
120
140

FIGURE 11. Codeword utilization, Kohonen self-organizing feature map net.

V. CONCLUSIONS

We have described a new method of vector quantization using neural networks. The training algorithm is based on the frequency-sensitive competitive learning method. We have compared this vector quantization procedure with the traditional LBG algorithm, the competitive learning algorithm, and the Kohonen self-organizing feature map.

Our results indicate that the FSCL method yield results which are comparable to both the LBG algorithm and the KSFM net for the vector quantization task. The methods were compared for a realistic task of encoding speech signals. We are presently investigating the applications of the FSCL method to classification problems in radar target identification and in speech recognition. The VQ algorithms are also being extended to the encoding of image data.



Codeword Utilization
FSCL Algorithm
128 Codewords
Itakura-Saito distortion : 0.3567
Entropy : 6.917
Percent Use
Codeword Index
0
1
2
3
4
5
6
20
40
60
80
100
120
140

FIGURE 12. Codeword utilization, frequency-sensitive competitive learning net.

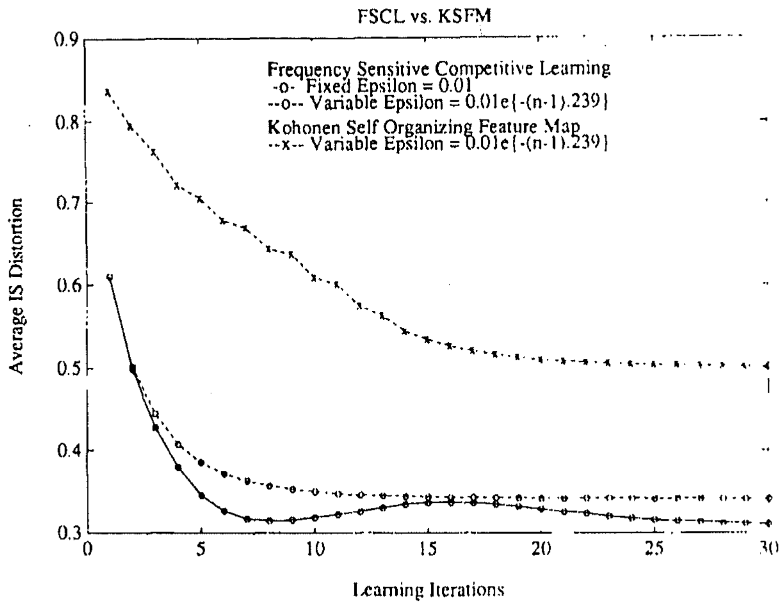

FSCL vs. KSFM
Frequency Sensitive Competitive Learning
-o- Fixed Epsilon = 0.01
--o-- Variable Epsilon = 0.01e{-(n-1).239}
Kohonen Self Organizing Feature Map
--x-- Variable Epsilon = 0.01e{-(n-1).239}
Average IS Distortion
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
Learning Iterations

FIGURE 13. Comparison of learning step size, KSFM and FSCL nets.

5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
0.5
1
1.5
2
2.5
2.9
Time (seconds)

FIGURE 14. Spectrogram of original sentence.

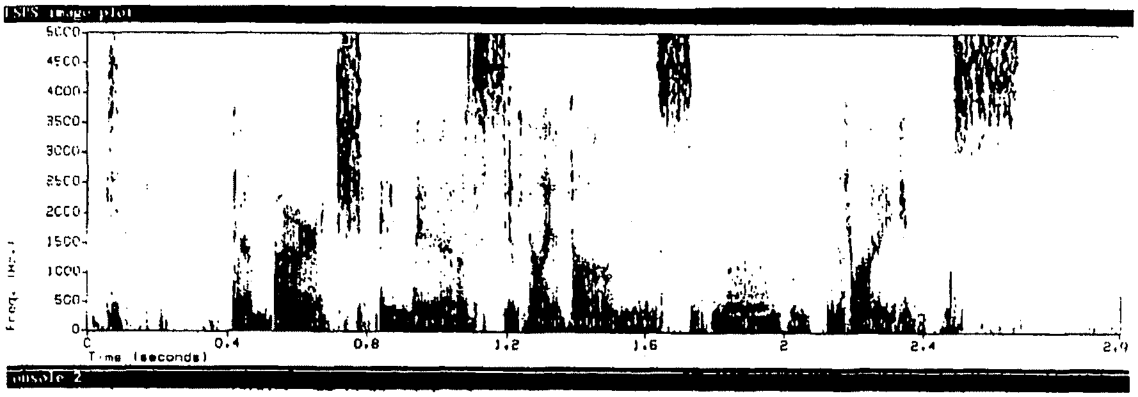

5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
0.4
0.8
1.2
1.6
2
2.4
2.9
Time (seconds)

FIGURE 15. Spectrogram of vector quantized LPC parameters of original sentence.

REFERENCES

Ahalt, S. C., Collins, S. A., Krishnamurthy, A. K., & Stewart, D. (1989). Closest vector selection in neural networks: Applications and implementation. (Tech. Rep. No. SPANN-1-89). Columbus, OH: Ohio State University.

Buzo, A., Gray, Jr., A. H., Gray, R. M., & Markel, J. D. (1980). Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **28**(5), 562–574.

Davidson, G. A., Cappello, P. R., & Gersho, A. (1988). Systolic architectures for vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **36**(10), 1651–1665.

DeSieno, D. (1988). Adding a conscience to competitive learning. In *IEEE International Conference on Neural Networks*, 1117–1124.

Elachi, C. (1987). *Introduction to the physics of remote sensing*. New York: Wiley Intersciences.

Entropic Speech, Inc. (1989). *Entropic signal processing systems*. Washington, DC: Author.

Gersho, A. (1982). On the structure of vector quantizers. *IEEE Transactions on Information Theory*, **28**(2), 157–166.

Gray, R. M. (1984). Vector quantization. *IEEE ASSP Magazine*, **1**(2), 4–29.

Grossberg, S. (1976a). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, 121–134.

Grossberg, S. (1976b). Adaptive pattern classification and universal recoding: Ii. Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, **23**, 187–202.

Grossberg, S. (1987). Competitive learning: From interactice activation to adaptive resonance. *Cognitive Science*, **11**, 23–63.

Hecht-Nielsen, R. (1988). Applications of counterpropagation networks. *Neural Networks*, **1**(2), 131–141.

Kohonen, T. (1984). Self-organization and associative memory. Berlin: Springer-Verlag.

Kohonen, T. (1988a). An introduction to neural computing. *Neural Networks*, **1**(1), 3–16.

Kohonen, T. (1988b). Learning vector quantization. In *Abstracts of the First Annual INNS Meeting* (p. 303). New York: Pergamon Press.

Kohonen, T. (1988c). Self-organization and associative memory (2nd ed.). Berlin: Springer-Verlag.

Linde, Y., Buzo, A., & Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, **28**(1), 84–95.

Lippmann, R. R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, **4**(2), 4–22.

Lloyd, S. P. (1982). Least-square quantization in PCM. *IEEE Transactions on Information Theory*, **28**(2), 129–137.

Makhoul, J., Roucos, S., & Gish, H. (1985). Vector quantization in speech coding. *Proceedings of the IEEE*, **73**(11), 1551–1588.

Matsuyama, Y. (1988). Vector quantization with optimized grouping and parallel distributed processing. *Journal of Neural Networks*.

Nasrabadi, N. & King, R. A. (1988). Image coding using vector quantization: A review. *IEEE Transactions on Communications*, **36**(8), 957–971.

Nasrabadi, N. M., & Feng, Y. (1988). Vector quantization of images based upon the Kohonen self-organizing feature maps. In *IEEE International Conference on Neural Networks* (pp. 1101–1108). San Diego: IEEE.

Naylor, J., & Li, K. P. (1988). Analysis of a neural network algorithm for vector quantization of speech parameters. In *Proceedings of the First Annual INNS Meeting* (p. 310). New York: Pergamon Press.

Netravali, A. N., & Mounts, F. W. (1980). Ordering techniques for facsimile coding: A review. *Proceedings of the IEEE*, **68**, 796–807.

O'Shaughnessy, D. (1987). *Speech communication: Human and machine*. Reading, MA: Addison-Wesley.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing*. Cambridge, MA: MIT Press.

Rumelhart, D. E., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, **9**, 75–112.

Shannon, C. E. (1960). Coding theorem for a discrete source with a fidelity criterion. In R. E. Machol (Ed.), *Information and decision processes*. New York, NY: McGraw-Hill.

Steels, L. (1988). Self-organization through selection. In *IEEE International Conference on Neural Networks* (pp. II55–II62). San Diego: IEEE.

Van den Bout, D. E., & Miller, III, T. K. (1989). TInMANN: The integer Markovian artificial neural network. In *Proceedings of the International Joint Conference on Neural Networks* (pp. II205–II211). Engelwood Cliffs, NJ: Erlbaum.

Winters, J. H., & Rose, C. (1989). Minimum distance automata in parallel networks for optimum classification. *Neural Networks*, **2**, 127–132.